Undecidable problems for propositional calculi with implication

Grigoriy V. Bokov
Department of Mathematical Theory of Intelligent Systems
Lomonosov Moscow State University
Moscow, Russian Federation
E-mail: bokov@intsys.msu.ru

February 4, 2015

Abstract

In this article, we deal with propositional calculi over a signature containing the classical implication $\to$ with the rules of modus ponens and substitution. For these calculi we consider few recognizing problems such as recognizing derivations, extensions, completeness, and axiomatizations. The main result of this paper is to prove that the problem of recognizing extensions is undecidable for every propositional calculus, and the problems of recognizing axiomatizations and completeness are undecidable for propositional calculi containing the formula $x \to (y \to x)$. As a corollary, the problem of derivability of a fixed formula A is also undecidable for all A. Moreover, we give a historical survey of related results.

1 Introduction

In 1946, Tarski [24] proposed to consider decision problems for a propositional calculus, which is defined as a finite set of propositional formulas over some signature with a finite set of rules of inference. Many important and interesting problems arise for these calculi. For example, *recognizing axiomatizations*, i.e., whether a given finite set of formulas constitutes (axiomatizes) an adequate axiom system for a propositional calculus, *recognizing extensions*, i.e., whether a given finite set of formulas derives all theorems of propositional calculus, *recognizing completeness*, i.e., whether a given finite set of theorems of propositional calculus constitutes an adequate axiom system for this calculus, and *recognizing derivations*, i.e., whether a given formula derives from a propositional calculus. In this paper we consider only propositional calculi with the rules of modus ponens and substitution.

The first undecidable problem for propositional calculi was found by Linial and Post in 1949 [15]. They proved the undecidability of recognizing completeness for the classical propositional calculus over the signature $\{\neg, \vee\}$. Note that Linial and Post gave sketch of proof, the full proof of their result was restored later by Davis [5, pp. 137–142] and Yntema [30]. The Linial and Post theorem is an example of the first undecidable propositional calculus, i.e., the problem of recognizing derivations is undecidable for this calculus. As a

corollary of this result, the problems of recognizing axiomatizations and extensions are also undecidable for the classical propositional calculus.

In 1963, Kuznetsov [14] proved the Linial and Post theorem for the intuitionistic calculus over the signature $\{\neg, \vee, \&, \to\}$. Moreover, he obtained a much stronger result, that the problem of recognizing completeness, as well as the problems of recognizing axiomatisations and extensions, is undecidable not only for the intuitionistic, but also for every superintuitionistic propositional calculus, i.e., a finitely axiomatizable extension of the intuitionistic calculus. Particularly, the Kuznetsov theorem implies that the intuitionistic propositional calculus contains undecidable propositional calculi.

Several constructions of undecidable propositional calculi have been obtained. Singletary in 1964 [25] constructed an undecidable propositional calculus over the signature $\{\neg, \to\}$. In 1965, Gladstone [6] and independently Ihrig [12] constructed propositional calculi for which the problem of recognizing derivations of formulas is of any required recursively enumerable degree of unsolvability. Note that Gladstone obtained the same result for every signature in which the implication is expressed. A much stronger result was obtained by Singletary in 1968 [26]. He constructed a pure implicational undecidable propositional calculus, i.e., calculus over the signature $\{\to\}$ whose axioms are derived from the axiom $x \to (y \to x)$.

Kuznetsov noticed in [14] that A. A. Markov (Jr.) in 1961 proposed to consider the same class of recognizing problems for the implicational propositional calculus. In this way, Harrop in 1964 [9] proved that the problem of recognizing completeness, as well as the problems of recognizing axiomatisations and extensions, is undecidable for every propositional calculus containing the formulas $x \to (y \to x)$ and $x \to x$. Independently, in 1972 Bollman and Tapia [3] by using Singletary constructions [26] proved the undecidability of the problem of recognizing extensions for the pure implicational fragment of the intuitionistic propositional calculus, i.e., the calculus with the following two axioms

$$x \to (y \to x), \text{ and}$$
$$(x \to (y \to z)) \to ((x \to y) \to (x \to z)).$$

In 1994, Marcinkowski [17] obtained a much stronger result: fix an implicational propositional tautology A that is not of the form $B \to B$ for some formula B, then the problem of recognizing extensions is undecidable for propositional calculus with the single axiom A. If we combine this with the Tarski result [27, p. 59], we obtain that the problem of recognizing extensions is undecidable for every finitely axiomatizable extension of the propositional calculus with axioms $x \to (y \to x)$ and $x \to (y \to ((x \to (y \to z)) \to z))$.

Some recent observations of related results were given in 2014 by Zolin [31] and Bokov [2]. Besides, an interesting observation was found by Chvalovský. He noted that the Linial and Post theorem for finitely represented superintuitionistic logics easily follows from Marcinkowski's construction in [17].

The aim of this paper is to prove that the problem of recognizing extensions is undecidable for all propositional calculus and to show that a derivation of the formula $x \to (y \to x)$ is sufficient for the undecidability of the problems of recognizing axiomatizations and completeness, i.e., every propositional calculus containing the formula $x \to (y \to x)$ has undecidable problems of recognizing axiomatizations and completeness. As a corollary, the problem, whether a fixed formula A is derivable from a given finite set of formulas by the rules of modus ponens and substitution, is also undecidable for all formula A, not only of the form $B \to B$ in contrast with the Marcinkowski result. Moreover, we consider a general

methods of proving that a recognizing problem of propositional calculi is undecidable, and give a historical survey of related results.

This paper is organized as follows. In the next section we introduce the basic terminology and notation, give a historical survey of known results, and state our main result. Section 3 is devoted to a reduction of undecidable problems for propositional calculi. In the first part of this section we give a historical survey of methods to prove the undecidability of a recognizing problems for propositional calculi, describe a general method and illustrate it by examples. Next, we recall what a tag system is and formally reduce the halting problem of tag systems to the derivation problem of propositional calculi. In Section 4 we prove our results. Finally, in Section 5 we give some concluding remarks and discuss further researches

2 Preliminaries and results

We begin with some notation. Let us consider the language consisting of an infinite set of propositional variables $\mathcal{V}$ and the signature Σ, i.e., a finite set of connectives. Letters x, y, z, u, p, etc., are used to denote propositional variables. Usually connectives are binary or unary such as $\neg$, $\vee$, $\wedge$, or $\to$.

Propositional formulas or Σ-*formulas* are built up from the signature Σ and propositional variables $\mathcal{V}$ in the usual way. For example, the following notations

$$x, \quad \neg A, \quad (A \vee B), \quad (A \wedge B), \quad (A \to B)$$

are formulas over the signature $\{\neg,\ \vee,\ \wedge,\ \to\}$. Capital letters A, B, C, etc., are used to denote propositional formulas. Throughout the paper, we will omit the outermost parentheses in formulas and parentheses assuming the customary priority of connectives.

Let Σ be a signature containing the binary connective of implication $\to$. By a *propositional calculus* or a Σ-*calculus* we mean a finite set P of Σ-formulas referred to as *axioms* together with two rules of inference:

1) *modus ponens*

$$A,\ A \to B\ \vdash\ B;$$

2) *substitution*

$$A\ \vdash\ \sigma A,$$

where σA is the substitution instance of A, i.e., the result of applying the substitution σ to the formula A.

Denote by $[P]$ the set of derivable (or provable) formulas of a calculus P. A *derivation* in P is defined from the axioms and the rules of inference in the usual way. The statement that a formula A is derivable from P is denoted by $P \vdash A$.

Let us introduce the following pre-order relation on the set of all propositional calculus. We write $P_1 \leq P_2$ (or, equivalently, $P_2 \geq P_1$) if each derivable formula of P_1 is also derivable from P_2, i.e., if $[P_1] \subseteq [P_2]$. We write $P_1 \sim P_2$ and say that two calculi P_1 and P_2 are *equivalent* if $[P_1] = [P_2]$. Finally, we write $P_1 < P_2$ if $[P_1] \subsetneq [P_2]$.

Now we formally define the problems of *recognizing derivations* (**Drv**), *extensions* (**Ext**), *axiomatizations* (**Axm**), and *completeness* (**Cmpl**) for a fixed Σ-calculus P_0:

(**Drv**) *given a calculus P, determine whether $P_0 \geq P$;*
(**Ext**) *given a calculus P, determine whether $P_0 \leq P$;*
(**Axm**) *given a calculus P, determine whether $P_0 \sim P$;*
(**Cmpl**) *given a calculus P such that $P \leq P_0$, determine whether $P_0 \leq P$.*

Denote by $\mathbf{Cl}_\Sigma$ the classical propositional calculus over a signature Σ, and by $\mathbf{Int}_\Sigma$ the intuitionistic propositional calculus over a signature Σ [13].

The previous results can be summarized as follows.

Theorem 2.1 (Linial and Post, 1949)**.** ***Axm***, ***Ext***, *and* ***Cmpl*** *are undecidable for* $\mathbf{Cl}_{\{\neg,\vee\}}$.

Theorem 2.2 (Kuznetsov, 1963)**.** *Fix a calculus $P_0 \geq \mathbf{Int}_{\{\neg,\vee,\wedge,\to\}}$, then* ***Axm***, ***Ext***, *and* ***Cmpl*** *are undecidable for P_0.*

Consider the intuitionistic implicational propositional calculus $\mathbf{Int}_{\{\to\}}$ with the set of axioms [10, p.69]:

$$\begin{array}{ll} (\mathrm{A}_1) & x \to (y \to x), \\ (\mathrm{A}_2) & (x \to (y \to z)) \to ((x \to y) \to (x \to z)). \end{array}$$

The classical implicational propositional calculus $\mathbf{Cl}_{\{\to\}}$ is obtained from $\mathbf{Int}_{\{\to\}}$ by adding the Peirce law $((x \to y) \to x) \to x$ [27, p.52].

Theorem 2.3 (Bollman and Tapia, 1964)**.** ***Ext*** *is undecidable for* $\mathbf{Int}_{\{\to\}}$.

Theorem 2.4 (Marcinkowski, 1994)**.** *Fix a $\{\to\}$-tautology A that is not of the form $B \to B$ for some formula B, then* ***Ext*** *is undecidable for the $\{\to\}$-calculus $\{A\}$.*

Since the implicational calculi $\mathbf{Cl}_{\{\to\}}$ and $\mathbf{Int}_{\{\to\}}$ can be axiomatized by the following single formulas, as shown by Łukasiewicz [16] and Meredith [18],

$$\begin{array}{rl} \mathbf{Cl}_{\{\to\}} & \sim \{((x \to y) \to z) \to ((z \to x) \to (u \to x))\} \\ \mathbf{Int}_{\{\to\}} & \sim \{((x \to y) \to z) \to (u \to ((y \to (z \to v)) \to (y \to v)))\} \end{array}$$

the following result also makes sense.

Corollary 2.5. ***Axm***, ***Cmpl*** *are undecidable for* $\mathbf{Cl}_{\{\to\}}$, *and* ***Ext*** *is undecidable for* $\mathbf{Cl}_{\{\to\}}$ *and* $\mathbf{Int}_{\{\to\}}$.

In 1930, Tarski [27] proved that every propositional calculus, which contains the formulas $x \to (y \to x)$ and $x \to (y \to ((x \to (y \to z)) \to z))$, can be axiomatized by a single formula. Since these formulas are derivable from $\mathbf{Int}_{\{\to\}}$, we have the following corollary of the Marcinkowski result.

Corollary 2.6. *Fix a calculus $P_0 \geq \mathbf{Int}_{\{\to\}}$, then* ***Ext*** *is undecidable for P_0.*

Theorem 2.7 (Bokov and Marcinkowski [1], 2014)**.** *Fix a calculus $P_0 \geq \mathbf{Int}_{\{\to\}}$, then* ***Axm*** *and* ***Cmpl*** *are undecidable for P_0.*

[1] According to the recent Chvalovský observation.

It is important to note that Corollary 2.6 and Theorem 2.7 was obtained quite a long time ago by Harrop [9].

Theorem 2.8 (Harrop, 1964)**.** *Fix a calculus $P_0 \geq \{x \to (y \to x),\ x \to x\}$, then **Axm**, **Ext**, and **Cmpl** are undecidable for P_0.*

Therefore, in order to prove the undecidability of the recognizing problem for a propositional calculus P_0, we must prove derivations of the formulas $x \to (y \to x)$ and $x \to x$ from P_0. In this paper we show that the second derivation, i.e., the derivation of the formula $x \to x$, is redundant to prove the undecidability of **Axm** and **Cmpl**. Indeed, as it was shown by Singletary in [26] the derivation of the formula $x \to (y \to x)$ is sufficient to construct an undecidable propositional calculus.

Theorem 2.9 (Singletary, 1968)**.** *There exists a propositional calculus $P_0 \leq \{x \to (y \to x)\}$ for which **Drv** is undecidable.*

Furthermore, we also prove that the derivation of the formula $x \to (y \to x)$ is redundant to prove the undecidability of **Ext**. Thus, our main result is the following theorem.

Theorem 2.10. *Fix a propositional calculus P_0, then*
(1) ***Ext** is undecidable for P_0;*
(2) ***Cmpl** is undecidable for P_0 if $P_0 \geq \{x \to (y \to x)\}$.*

As corollary, we have the undecidability of the problem of recognizing axiomatizations.

Corollary 2.11. *Fix a calculus $P_0 \geq \{x \to (y \to x)\}$, then **Axm** is undecidable for P_0.*

Moreover, if we take in the theorem 2.10 the propositional calculus $P_0 = \{A\}$ for a Σ-formula A, then we obtain the undecidability of problem of derivability.

Corollary 2.12. *Fix a signature $\Sigma \supseteq \{\to\}$ and a Σ-formula A, then the following problem is undecidable:*

$$\textit{given a } \Sigma\textit{-calculus } P\textit{, determine whether } P \vdash A.$$

Particularly, this holds for a formula A of the form $B \to B$ for some formula B in contrast with Theorem 2.4.

3 Reduction of undecidable problems

The typical method of proving a problem to be undecidable is a reduction of famous undecidable problem to this problem. In order to do this, it is sufficient to transform instances of an undecidable problem into instances of the new problem so that if a solution to the new problem were found, it could be used to decide the undecidable problem. Since we already know that no method can decide the old problem, no method can decide the new problem also.

3.1 Historical survey

One of the first problems to be proved undecidable is the halting problem of Turing machines [28]. For example, Harrop [9] and Hughes [11] simulated Turing machines by implicational propositional calculi and reduced the halting problem to the decision problem of a partial implicational propositional calculus. Note that Hughes used only formulas contain at most two distinct variable symbols. But in some cases it is more convenient to reduce other undecidable problems.

Often decision problems for propositional calculi are associated with the word problem for semi-Thue systems. So, by a reduction of semi-Thue systems Yntema [30] proved the undecidability of the completeness problem, Gladstone [6] and independently Ihrig [12] constructed calculi for which the problem of derivability of formulas is of any required recursively enumerable degree of unsolvability, Singletary [26] constructed an undecidable implicational calculus, Boolman and Tapia [3] proved that it is impossible to algorithmically determine of an arbitrarily given partial propositional calculus whether or not the deduction theorem holds.

Numerous results were obtained on a simulation of Post normal system [21] with the undecidable halting problem. For example, Linial and Post [15] noted that the undecidability of the completeness problem for the classical propositional calculus can be proved by a reduction of normal system introduced in [22]. In the same way Harrop [8] proved existence of undecidable propositional calculus, Ratsa [23] proved the undecidability of the expressibility problem for modal logics. Recently, Zolin [31] obtained the Kuznetsov's results by a reduction of tag systems, i.e., a simple form of Post normal systems. A reduction of the halting problem of tag systems has been proposed by Bokov [1] for a proof of the Linial and Post theorem and improved in [2] for a proof of the undecidability of some recognizing problems for propositional calculi with implication.

The above results are combined by using the halting condition of some computational machine such as Turing machine, semi-Thue system, or Post normal system. Another example of these machines is counter machines such as Minsky machines [20]. Chagrov [4] used Minsky machines to prove the undecidability of some problems of modal logics.

Nevertheless, there are reductions of other undecidable problems, not only the halting problem of some computational machine. So, Kuznetsov [14] devised a special calculus of primitive recursive functions, Marcinkowski [17] investigated the entailment problem for first-order Horn clauses.

3.2 General method and examples

In this section we describe a general method of reduction for undecidable problems of propositional calculi. First, for a given propositional calculus we must to fix

- *a model of computation* that is equivalent in its computational power to Turing machines, such as semi-Thue systems, Post normal systems, tag systems, or Minsky machines, and
- *a procedure of encoding* that allows to encode operations of computation used in this model and their respective costs by a formulas of the propositional calculus.

Next, we must to simulate the model of computation by the inference process of the propositional calculus.

As an example, let us consider an abstract computational machine T, which deals with words over a finite alphabet $\mathcal{A}$. Operations of this machine are a finite set R of pairs of words over $\mathcal{A}$.

A computation of the machine T on an input word ξ is a sequence of words $\lambda_0 = \xi, \lambda_1, \ldots$ such that every pair $(\lambda_i, \lambda_{i+1})$ is a instance of some operation from R for all $i \geq 0$. Note that computations must be deterministic. We write $\xi \overset{T}{\Longmapsto} \zeta$ if there is a computation $\lambda_0, \lambda_1, \ldots, \lambda_n$, $n > 0$, such that $\lambda_0 = \xi$, $\lambda_n = \zeta$.

The halting condition of the machine T is a finite set H of words over $\mathcal{A}$. We say that the machine T halts on input ξ if the computation of T on ξ reaches a word from H, i.e., $\xi \overset{T}{\Longmapsto} \zeta$ for some $\zeta \in H$.

Next, let us consider propositional calculi with modus ponens and substitution. Assume that we want to prove an undecidability of the following recognizing problem: fix a class of propositional calculi $\mathcal{P}$ and a propositional calculus P_0, whether a given calculus $P \in \mathcal{P}$ contains P_0, i.e., $P \geq P_0$? In order to prove the undecidability of this problem, we fix a machine T with the undecidable halting problem. Next, we encode words over $\mathcal{A}$ and construct a propositional calculus for the machine T such that derivations of the codes of words simulates a computation of T on them.

More precisely, let $\overline{\alpha}$ be the code of a word $\alpha \in \mathcal{A}^*$, and $\overline{\xi} \to \overline{\zeta}$ the code of a operation $(\xi, \zeta) \in R$. Usually, the code of any instance of operation (ξ, ζ) can be obtained from the code $\overline{\xi} \to \overline{\zeta}$ by substitution. We must construct a propositional calculus P_T such that

1. the computation of machine T simulates as follows:

$$\xi \overset{T}{\Longmapsto} \zeta \quad \text{iff} \quad P_T,\ \overline{\xi} \ \vdash\ \overline{\zeta};$$

2. the halting condition of T defines as follows:

$$T \textit{ halts on } \xi \quad \text{iff} \quad P_T,\ \overline{\xi} \ \vdash\ P_0;$$

3. a calculus obtained from P_T by adding the axiom $\overline{\xi}$ is in $\mathcal{P}$.

Then we obtain that the problem of recognizing, whether $P \geq P_0$ for a given calculus $P \in \mathcal{P}$, is undecidable, since otherwise the halting problem for T is decidable.

In next sections we describe a process of reduction of the undecidable halting problem for abstract computational machines to a recognizing problem for propositional calculi in more details. For this reason, we take the tag system introduced by Post [21] as an example of computational machine and consider propositional calculi over the signature Σ such that $\{\to\} \subseteq \Sigma$. For a given Σ-calculus P_0, tag system T and a word ξ, we will effectively construct a Σ-calculus $P_{T,P_0,\xi}$ such that T halts on the input word ξ if and only if $P_0 \leq P_{T,P_0,\xi}$. Then the proof of Theorem 2.10 is immediately following from the undecidability of the halting problem [19].

First let us recall the notion of a tag system.

3.3 Tag systems

Let $\mathcal{A}$ be a finite alphabet of letters $a_1, \ldots, a_m$. By $\mathcal{A}^*$ denote the set of all words over $\mathcal{A}$, including the empty word. For $\alpha \in \mathcal{A}^*$, denote by $|\alpha|$ the length of the word α.

Definition 3.1 (Post, [21]). A *tag system* is a triple $T = \langle \mathcal{A}, \mathcal{W}, d \rangle$, where $\mathcal{A} = \{a_1, \ldots, a_m\}$ is a finite alphabet of m symbols, $\mathcal{W} = \{\omega_1, \ldots, \omega_m\} \subseteq \mathcal{A}^*$ is a set of m words, and $d \in \mathbb{N}$ is a *deletion number*. Each words ω_i is associated to the letters a_i: $a_1 \to \omega_1, \ldots, a_m \to \omega_m$.

We say that T is applicable to a word $\alpha \in \mathcal{A}^*$ if $|\alpha| \geq d$. The application of T to a word $\alpha \in \mathcal{A}^*$ is defined as follows. Examine the first letter of the word α. If it is a_i then

1. remove the first d letters from α, and
2. append to its end the word ω_i.

Perform the same operation on the resulting word, and repeat the process as long as the resulting word has d or more letters. To be precise, if $\alpha = a_i\beta\gamma$, $|\beta| = d - 1$, and $\gamma \in \mathcal{A}^*$, then T produces the word $\gamma\omega_i$ from the word $a_i\beta\gamma$. Denote this production by $a_i\beta\gamma \overset{T}{\longmapsto} \gamma\omega_i$. We write $\alpha \overset{T}{\Longmapsto} \beta$ if there are words $\gamma_1, \ldots, \gamma_n$, $n \geq 1$, such that $\alpha = \gamma_1$, $\beta = \gamma_n$, and $\gamma_i \overset{T}{\longmapsto} \gamma_{i+1}$ for all $1 \leq i \leq n - 1$.

Define the halting problem of tag systems. We say that a tag system T *halts* on a word $\alpha \in \mathcal{A}^*$ and write this as $T(\alpha) \downarrow$ if there exists a word $\beta \in \mathcal{A}^*$ such that $\alpha \overset{T}{\Longmapsto} \beta$ and T is not applicable to β, i.e. $|\beta| < d$. The *halting problem* for a fixed tag system T is, given any word $\alpha \in \mathcal{A}^*$, to determine whether T halts on α.

Theorem 3.2 (Minsky, [19]). *There is a tag system T for which the halting problem is undecidable.*

Moreover, Wang [29] showed that this holds even for some tag system T with $d = 2$ and $1 \leq |\omega_i| \leq 3$ for all $1 \leq i \leq m$. For this reason, throughout the paper we will assume that all words ω_i are nonempty.

3.4 Encoding of letters and words

Let $\mathcal{A}$ be a finite set $\{a_1, \ldots, a_m\}$ as above. The set of all nonempty words over $\mathcal{A}$ is denoted by $\mathcal{A}^+$. We encode letters and words over $\mathcal{A}$ as one variable $\{\to\}$-formulas. In order to simulate a tag system over alphabet $\mathcal{A}$ correctly, this encoding must be an injective function between words over $\mathcal{A}$ and their codes. As shown in [1], a word-to-formula encoding is related with difficulties of derivation of a code of one word from a code of other word. Below we show that it is more convenient to encode a word as a set of formulas. Moreover, we give a one-to-one (bijective) encoding between words over $\mathcal{A}$ and their codes.

Fix a one-variable $\{\to\}$-formula $\hat{x}$. As an example of $\hat{x}$ may be the formula $x \to x$ or $x \to (x \to x)$. Note that $\hat{x}$ is an arbitrary $\{\to\}$-formula with a single variable x. For future use we introduce a shortcut for the following formula of two variables:

$$x \circ y := ((\hat{y} \to \hat{y}) \to \hat{y}) \to (\hat{x} \to ((\hat{y} \to \hat{y}) \to \hat{y})).\ ^{2}$$

It is obvious that $x \circ y$ is a substitution instance of the axiom $x \to (y \to x)$. The following lemma is needed for the sequel.

Lemma 3.3. *$x \circ y$ and $(x \circ y) \to z$ are not unifiable.*

[2]Note that $\hat{y}$ is the substitution instance of $\hat{x}$ with replacing x by y.

The proof is straightforward and left to the reader.

Now we define the notion of code of a letter. First, let us fix a unique variable p. Then the *code* of a letter $a_i \in \mathcal{A}$, for $1 \leq i \leq m$, is a formula:

$$\overline{a_i} := \big(\underbrace{p \to (p \to \ldots (p}_{i} \to p))\big) \circ p. \tag{1}$$

Since $x \circ y$ is a substitution instance of the axiom $x \to (y \to x)$, we have the following lemma.

Lemma 3.4. *$x \to (y \to x) \vdash \overline{a}$, for every letter $a \in \mathcal{A}$.*

In order to encode a word, i.e. finite sequence of letter, we must define an operation of concatenation for letters. For this reason, we introduce a shortcut $x \cdot y$ as an abbreviation for the formula $((x \to x) \to x) \circ y$. Thus we come to the following definition.

Definition 3.5. (Zolin, [31]) An *alphabetic formula* over the alphabet $\mathcal{A}$, or an *$\mathcal{A}$-formula* for short, is an arbitrary $\{\cdot\}$-formula over the codes of letters from $\mathcal{A}$. Formally, $\overline{a}$ is a $\mathcal{A}$-formula for each letter $a \in \mathcal{A}$, and if A, B are $\mathcal{A}$-formulas then so is $A \cdot B$.

An example of an alphabetic formula is $(\overline{a} \cdot (\overline{c} \cdot \overline{e})) \cdot (\overline{e} \cdot \overline{a})$. It is easily seen that every $\mathcal{A}$-formula is associated with a word over $\mathcal{A}$. To every $\mathcal{A}$-formula A we associate its $word(A) \in \mathcal{A}^+$ by induction: $word(\overline{a}) := a$ for each letter $a \in \mathcal{A}$, and $word(A \cdot B) := word(A)word(B)$. For example, the $\mathcal{A}$-formulas $(\overline{a} \cdot \overline{e}) \cdot (\overline{c} \cdot \overline{a})$ and $(\overline{a} \cdot (\overline{e} \cdot \overline{c})) \cdot \overline{a}$ are associated with the same word $aeca$.

Let us introduce some notation that will be useful later. Given a formula A, denote by A^* the set of all substitution instances of A. Similarly, given a set M of formulas, denote by M^* the set

$$M^* := \bigcup_{A \in M} A^*.$$

We call two formulas A and B *unifiable* if $A^* \cap B^* \neq \emptyset$. For example, formulas $x \to (y \to z)$ and $(y \to z) \to x$ are unifiable, but formulas $x \to (y \to x)$ and $(y \to x) \to x$ are not unifiable.

Lemma 3.6. *No two distinct $\mathcal{A}$-formulas are unifiable.*

Proof. By induction on the definition of an $\mathcal{A}$-formula A.
Let A be the code of a letter $a_i \in \mathcal{A}$. There are two cases:

1) If B is the code of a letter $a_j \in \mathcal{A}$, then $i \neq j$. Denote by C_i the following formula

$$\underbrace{p \to (p \to \ldots (p}_{i} \to p)).$$

Then A is the formula $C_i \circ p$ and B is the formula $C_j \circ p$. Since C_i and C_j are not unifiable for $i \neq j$, we conclude that A and B are not unifiable.

2) Let B is a formula $B_1 \cdot B_2$ for some $\mathcal{A}$-formulas B_1 and B_2. Then A is the formula $C_i \circ p$ and B is the formula $((B_1 \to B_1) \to B_1) \circ B_2$. Since the formulas C_i and $(x \to x) \to x$ are not unifiable for all i, $1 \leq i \leq m$, we have that A and B are not unifiable, .

Now let $A = A_1 \cdot A_2$ for some $\mathcal{A}$-formulas A_1 and A_2, so it can be assumed that $B = B_1 \cdot B_2$ for some $\mathcal{A}$-formulas B_1 and B_2. If A, B are unifiable, then also A_1, B_1 and A_2, B_2 are unifiable. By induction hypothesis, $A_1 = B_1$ and $A_2 = B_2$. Hence, $A = B$.

This completes the proof of the lemma. □

Finally, we define the *code* of a word $\alpha \in \mathcal{A}^+$ as the finite set $\overline{\alpha}$ consisting of all $\mathcal{A}$-formulas associated with the word α. Formally,

$$\overline{\alpha} := \{A \mid A \text{ is a } \mathcal{A}\text{-formula such that } word(A) = \alpha\}.$$

Note that the code of a letter $a \in \mathcal{A}$ is the formula defined as in (1), but the code of a single-letter word $a \in \mathcal{A}^+$ is the set consisting of the code of letter a. Throughout this paper, we will use the same notation for the code of a letter and the code of a single-letter word.

As an example, $\{\overline{a} \cdot \overline{c}\}$ is the code of the word ac, $\{\overline{a} \cdot (\overline{c} \cdot \overline{e}),\ (\overline{a} \cdot \overline{c}) \cdot \overline{e}\}$ is the code of the word ace, and $\{\overline{a} \cdot (\overline{c} \cdot (\overline{e} \cdot \overline{c})),\ \overline{a} \cdot ((\overline{c} \cdot \overline{e}) \cdot \overline{c}),\ ((\overline{a} \cdot \overline{c}) \cdot (\overline{e} \cdot \overline{c})),\ (\overline{a} \cdot (\overline{c} \cdot \overline{e})) \cdot \overline{c},\ ((\overline{a} \cdot \overline{c}) \cdot \overline{e}) \cdot \overline{c}\}$ is the code of the word $acec$, where $a, c, e \in \mathcal{A}$.

Since every $\mathcal{A}$-formula is a substitution instance of the axiom $x \to (y \to x)$, we have the following generalization of Lemma 3.4.

Lemma 3.7. *$x \to (y \to x) \vdash \overline{\alpha}$, for every word $\alpha \in \mathcal{A}^+$.*

Similarly, we call two codes $\overline{\alpha}$ and $\overline{\gamma}$ *unifiable* if $\overline{\alpha}^* \cap \overline{\gamma}^* \neq \emptyset$. Lemma 3.6 implies:

Corollary 3.8. *No two distinct codes are unifiable.*

Now we introduce the following convention. In order to simplify a notation of formulas, we will use an abbreviation $\overline{\alpha}$ for some word $\alpha \in \mathcal{A}^+$ as a part of formulas. For example, a formula $\overline{\alpha} \to x$ is a shortcut for the following set of formulas

$$\{A \to x \mid A \in \overline{\alpha}\},$$

and a formula $\overline{\alpha} \cdot x \to x \cdot \overline{\beta}$ is a shortcut for the set

$$\{(A \cdot x) \to (x \cdot B) \mid A \in \overline{\alpha},\ B \in \overline{\beta}\}.$$

Note that all alphabetic formulas are one-variable formulas with the same variable p, so we substitute the same formula in different occurrences of alphabetic formulas. As an example of this substitution let us consider a formula A of the form

$$\overline{\alpha}[p] \cdot x \to x \cdot \overline{\beta}[p],$$

where square brackets denote a dependence on variables or subformulas. Then a substitution instance of A is any formula of the form

$$\overline{\alpha}[B] \cdot C \to C \cdot \overline{\beta}[B]$$

for some formulas B and C.

3.5 Simulation of tag systems

For a given tag system T we construct a propositional $\{\to\}$-calculus P_T such that the derivation of codes of words in P_T simulates productions of words in T.

Let $T = \langle \mathcal{A}, \mathcal{W}, d \rangle$, where $\mathcal{A} = \{a_1, \ldots, a_m\}$, $\mathcal{W} = \{\omega_1, \ldots, \omega_m\}$, and $d \in \mathbb{N}$. Recall that all ω_i are assumed to be nonempty. Denote by P_T a $\{\to\}$-calculus with the following groups of axioms.

Productions of the tag system T:

$$\begin{array}{ll} (\mathrm{T}_1) & \overline{a_i\alpha}\cdot x \to x\cdot\overline{\omega_i} \\ (\mathrm{T}_2) & \overline{a_i\alpha}\to\overline{\omega_i} \end{array} \quad \text{for all } \alpha\in\mathcal{A}^+,\ |\alpha|=d-1,\ 1\le i\le m;$$

Transformation rules:

$$\begin{array}{ll} (\mathrm{R}_1) & x\cdot(y\cdot z)\to(x\cdot y)\cdot z \\ (\mathrm{R}_2) & (x\cdot y)\cdot z\to x\cdot(y\cdot z) \\ (\mathrm{R}_3) & (x\cdot(y\cdot z))\cdot u\to((x\cdot y)\cdot z)\cdot u \\ (\mathrm{R}_4) & ((x\cdot y)\cdot z)\cdot u\to(x\cdot(y\cdot z))\cdot u \end{array}$$

Define two subsystems of the calculus P_T:

$$\mathrm{T} := \mathrm{T}_1\cup\mathrm{T}_2,\quad \mathrm{R} := \mathrm{R}_1\cup\mathrm{R}_2\cup\mathrm{R}_3\cup\mathrm{R}_4.$$

Since they are rather weak and not even capable to derive $A\to C$ from $A\to B$ and $B\to C$, we introduce the following useful notation: $P\vdash A\Rightarrow B$ if and only if there are formulas $C_0=A,\ C_2,\ \ldots,\ C_{n-1},\ C_n=B$, $n\ge 0$, such that $P\vdash C_i\to C_{i+1}$ for all $0\le i\le n-1$.

Since every formula $A\cdot B$ is a substitution instance of the axiom $x\to(y\to x)$, we have the following lemma.

Lemma 3.9. $P_T\le\{x\to(y\to x)\}$.

Now we prove some properties of the calculus P_T.

3.5.1 Derivability of the T-productions

Here we show that the calculus P_T can "simulate" productions of the tag system T. At the beginning let us prove auxiliary lemmas.

Lemma 3.10. $\mathrm{R}\vdash A\Rightarrow\overline{\alpha}$, *for all* $\alpha\in\mathcal{A}^+$ *and* $A\in\overline{\alpha}$.

Proof. Let $\alpha=a_1\ldots a_n$. Since all axioms in R are invertible, i.e., $B\to A\in\mathrm{R}$ whenever $A\to B\in\mathrm{R}$, it is sufficient to prove that

$$\mathrm{R}\vdash A\Rightarrow\overrightarrow{\alpha},$$

where $\overrightarrow{\alpha}$ is the following formula $\overline{a_1}\cdot(\overline{a_2}\cdot\ldots\cdot(\overline{a_{n-1}}\cdot\overline{a_n}))$.

Without loss of generality it can be assumed that $n\ge 3$. We split up the proof into two steps. First, we will show that there exists an alphabetic formula B such that

$$\mathrm{R}\vdash A\Rightarrow B\cdot\overrightarrow{\xi},$$

where $\xi\in\mathcal{A}^+$ and $\alpha=\beta\xi$, for $\beta=word(B)$. Next, we will prove that

$$\mathrm{R}\vdash B\cdot\overrightarrow{\xi}\Rightarrow\overrightarrow{\beta\xi}$$

by induction on $|\beta|$.

Proof of the first step: Since $n \geq 3$, there is an integer $k \geq 1$, nonempty words $\alpha_1, \ldots, \alpha_k, \xi$ and alphabetic formulas $A_1, \ldots, A_k$ such that $\alpha = \alpha_1 \ldots \alpha_k \xi$,

$$A = A_1 \cdot (A_2 \cdot \ldots \cdot (A_k \cdot \overrightarrow{\xi})),$$

and $word(A_i) = \alpha_i$, for $1 \leq i \leq k$. Denote by B the following formula $(((A_1 \cdot A_2) \cdot A_3) \cdot \ldots \cdot A_k)$, then we have

$$\mathrm{R} \vdash A \Rightarrow B \cdot \overrightarrow{\xi}$$

by a multiple application of axiom (R_1).

Proof of the second step: By induction on $|\beta|$, where $\beta = word(B)$. If $|\beta| = 1$, then the formulas $B \cdot \overrightarrow{\xi}$ and $\overrightarrow{\beta\xi}$ are identical.

Now let $|\beta| \geq 2$, so there is an integer $m \geq 1$, a letter $a \in \mathcal{A}$, and nonempty words $\beta_1, \ldots, \beta_m$ such that $\beta = \beta_1 \ldots \beta_m a$ and

$$B = B_1 \cdot (B_2 \cdot \ldots \cdot (B_m \cdot \overline{a})),$$

for some alphabetic formulas $B_1, \ldots, B_m$ such that $word(B_i) = \beta_i$, $1 \leq i \leq m$. Denote by C the following formula $(((B_1 \cdot B_2) \cdot B_3) \ldots \cdot B_m)$. Then we derive in R:

$$B \cdot \overrightarrow{\xi} \quad \stackrel{(\mathrm{R}_3)}{\Longrightarrow} \quad (C \cdot \overline{a}) \cdot \overrightarrow{\xi} \quad \stackrel{(\mathrm{R}_2)}{\longrightarrow} \quad C \cdot \overrightarrow{a\xi} \quad \stackrel{\mathrm{IH}}{\Longrightarrow} \quad \overrightarrow{\beta\xi}$$

where the first derivation is a multiple application of axiom (R_3), the second derivation is a single application of axiom (R_2), and the last derivation uses induction hypothesis for the word γ such that $\gamma = word(C)$. Note that γ is exactly the word $\beta_1 \ldots \beta_m$. The lemma is proved. □

Lemma 3.11. *If $\xi \stackrel{T}{\longmapsto} \zeta$ then $P_T \vdash \overline{\xi} \Rightarrow \overline{\zeta}$, for all $\xi, \zeta \in \mathcal{A}^+$.*

Proof. Since T is applicable to ξ, we have $|\xi| \geq d$. Therefore, $\xi = a_i \alpha \beta$ and $\zeta = \beta \omega_i$, where $|\alpha| = d - 1$ and $|\beta| \geq 0$.

If $|\beta| = 0$, then $P_T \vdash \overline{\xi} \to \overline{\zeta}$ by the axiom (T_2).

Let $|\beta| > 0$, so we derive in P_T:

$$\overline{\xi} \quad \stackrel{\mathrm{L}}{\Longrightarrow} \quad \overline{a_i\alpha} \cdot \overline{\beta} \quad \stackrel{(\mathrm{T}_1)}{\longrightarrow} \quad \overline{\beta} \cdot \overline{\omega_i} \quad \stackrel{\mathrm{L}}{\Longrightarrow} \quad \overline{\zeta}$$

where the first and last derivations are due to Lemma 3.10, and the second derivation is the substitution instance of the axiom (T_1). The lemma is proved. □

Corollary 3.12. *If $\xi \stackrel{T}{\Longmapsto} \zeta$ then $P_T \vdash \overline{\xi} \Rightarrow \overline{\zeta}$, for all $\xi, \zeta \in \mathcal{A}^+$.*

The proof is trivial by definition of the tag system.

3.5.2 Production of the P_T-derivations

Here we show that the tag system T can produce, on the input word, the words whose codes have derivations in P_T. As a preliminary let us introduce some notation and prove auxiliary lemmas.

Given $\alpha \in \mathcal{A}^+$, denote by T_α the set of all $\mathcal{A}$-formulas whose words have productions of the tag system T on the input word α:

$$T_\alpha = \{A \mid A \text{ is a } \mathcal{A}\text{-formula such that } \alpha \stackrel{T}{\Longmapsto} word(A)\}.$$

It is clear that $\overline{\alpha} \subseteq T_\alpha$ for all $\alpha \in \mathcal{A}^+$.

Lemma 3.13. *$P_T^* \cap T_\alpha^* = \emptyset$, for all $\alpha \in \mathcal{A}^+$.*

The proof is trivial by application of Lemma 3.3.

For any propositional calculus P, denote by $\langle P \rangle$ the set of propositional formulas obtained from P by applying modus ponens and substitution once:

$$\langle P \rangle := \{B \mid A, A \to B \in P \text{ for some formula } A\} \cup \{\sigma A \mid A \in P \text{ and } \sigma \text{ is a substitution}\}.$$

Furthermore, let $\langle P \rangle_0 = P$ and

$$\langle P \rangle_{n+1} = \langle \langle P \rangle_n \rangle$$

for $n \geq 0$. It follows easily that $\langle P \rangle_n \subseteq \langle P \rangle_{n+1}$ for all $n \geq 0$ and the set $[P]$ of all derivable formulas of the calculus P can be represented as

$$[P] = \langle P \rangle_\infty = \bigcup_{n \geq 0} \langle P \rangle_n .$$

Let A be a formula derivable from P. We say that A has the *derivation height* n, if $A \in \langle P \rangle_n$ and $A \notin \langle P \rangle_{n-1}$.

The following theorem describes formulas derivable from the calculus P_T and the code of a nonempty word $\alpha \in \mathcal{A}^+$.

Lemma 3.14. *$[P_T \cup \overline{\alpha}] = P_T^* \cup T_\alpha^*$ for all $\alpha \in \mathcal{A}^+$.*

Proof. It is evident that

$$P_T^* \cup T_\alpha^* \subseteq [P_T \cup \overline{\alpha}]$$

by Lemma 3.10 and Corollary 3.12, so we only prove by induction on the derivation height $n \geq 0$ that

$$\langle P_T \cup \overline{\alpha} \rangle_n \subseteq P_T^* \cup T_\alpha^*.$$

If $n = 0$, then $\langle P_T \cup \overline{\alpha} \rangle_0 = P_T \cup \overline{\alpha}$. Clearly, $\overline{\alpha} \subseteq T_\alpha^*$ and all axioms of P_T are in P_T^*.

Let the induction assumption be satisfied for some $n \geq 1$. Since the right-hand side of the inclusion is closed under substitution, we only consider the case of a formula B obtained by modus ponens from some formulas A, $A \to B \in \langle P_T \cup \overline{\alpha} \rangle_n$. By induction hypothesis,

$$\langle P_T \cup \overline{\alpha} \rangle_n \subseteq P_T^* \cup T_\alpha^*.$$

It is easily shown that $P_T^* \cap T_\alpha^* = \emptyset$ due to Lemma 3.13. Hence either A or $A \to B$ are in T_α^*, since otherwise A is both a substitution instance of $x \circ y$ and $x \circ y \to z$, which is impossible by Lemma 3.3. If $A \to B \in T_\alpha^*$, then A is a substitution instance of the formula $(y \to y) \to y$. However, $A \in P_T^* \cap T_\alpha^*$, which is impossible, because all formulas in P_T^* and T_α^* are not unifiable with $(y \to y) \to y$. Therefore, $A \in T_\alpha^*$ and $A \to B \in P_T^*$.

Now we show that $B \in T_\alpha^*$. Since $A \in T_\alpha^*$, then $A \in \overline{\gamma}^*$ for some word $\gamma \in \mathcal{A}^+$ such that $\alpha \overset{T}{\Longmapsto} \gamma$. Note that $A \to B$ is a substitution instance of some axiom in P_T, so we need to consider the following two cases.

Case 1. $A \to B$ is a substitution instance of an axiom in T. Then A is a substitution instance of the formula $\overline{a_i \alpha_1} \cdot C$ or $\overline{a_i \alpha_1}$ for some letter $a_i \in \mathcal{A}$, a word $\alpha_1 \in \mathcal{A}^*$ with $|\alpha_1| = d - 1$, and a formula C. Since $A \in \overline{\gamma}^*$, we have that C is a alphabetic formula.

Therefore, by Lemma 3.6 there is a unique word $\alpha_2 \in \mathcal{A}^*$ such that $\gamma = a_1\alpha_1\alpha_2$. It is clear that $\alpha_2 = word(C)$ and B is the substitution instance of the alphabetic formula $C \cdot \overline{\omega_i}$ or $\overline{\omega_i}$. Thus, $B \in \overline{\eta}^*$ for $\eta = \alpha_2\omega_i$ and $\gamma \stackrel{T}{\longmapsto} \eta$.

Case 2. $A \to B$ is a substitution instance of an axiom in R. Since the formula A is a substitution instance of an alphabetic formula $C \in \overline{\gamma}$ and the set of alphabetic formulas $\overline{\gamma}$ is closed under application modus ponens and the axioms R, we have that also $B \in \overline{\gamma}^*$.

These cases exhaust all possibilities and so we have that $B \in \overline{\eta}^*$ for some word $\eta \in \mathcal{A}^*$ such that $\gamma \stackrel{T}{\Longmapsto} \eta$. Hence $B \in T^*_\alpha$, since $\alpha \stackrel{T}{\Longmapsto} \gamma$ by induction hypothesis. The proof is completed. □

Now we prove that the code of each nonempty word over $\mathcal{A}$ derivable from P_T and $\overline{\alpha}$ is the code of a word produced from α by the tag system T.

Corollary 3.15. *If $P_T \vdash \overline{\xi} \Rightarrow \overline{\zeta}$ then $\xi \stackrel{T}{\Longmapsto} \zeta$, for all $\xi, \zeta \in \mathcal{A}^+$.*

Proof. Let $P_T \vdash \overline{\xi} \Rightarrow \overline{\zeta}$, so $\overline{\zeta} \in [P_T \cup \overline{\xi}]$. Then $\overline{\zeta} \in P^*_T \cup T^*_\xi$ by Lemma 3.14. Since $\overline{\zeta} \notin P^*_T$ due to Lemma 3.3, we have that $\overline{\zeta} \in T^*_\xi$ and so $\xi \stackrel{T}{\Longmapsto} \zeta$ by definition of the set T_ξ. The lemma is proved. □

3.5.3 Halting condition

Above we shown that derivations in the propositional calculus P_T can simulate productions in the tag system T. Now we describe how to perform the halting condition of tag system T on input words. For this reason, we consider a propositional calculus P_0 and the following group of axioms.

The halting condition for the tag system T:

(H) $\quad \overline{\alpha} \to A \quad$ for all $\alpha \in \mathcal{A}^+,\ |\alpha| < d,\ A \in P_0$.

Denote by P_{T,P_0} the calculus $P_T \cup \mathrm{H}$, and by $P_{T,P_0,\xi}$ the calculus $P_T \cup \mathrm{H} \cup \overline{\xi}$. Let the tag system T halts on the input word ξ, we take the minimal $n \geq 0$ such that $\langle P_{T,P_0,\xi} \rangle_n$ contains at least one substitution instance of element of the code for some word $\zeta \in \mathcal{A}^+$ with $|\zeta| < d$:

$$N_\xi := \min \left\{ n \geq 0 \mid \overline{\zeta}^* \cap \langle P_{T,P_0,\xi} \rangle_n \neq \emptyset,\ \text{for some } \zeta \in \mathcal{A}^+ \text{ with } |\zeta| < d \right\}.$$

If T does not halt, then we put $N_\xi := \infty$. We have the following generalization of Lemma 3.14.

Lemma 3.16. $\langle P_{T,P_0,\xi} \rangle_{N_\xi} \subseteq P^*_{T,P_0} \cup T^*_\xi$.

Proof. Clearly, it is sufficient to consider the case of the proof of Lemma 3.14 for which $A \to B$ is a substitution instance of axioms (H). But this case is impossible, since otherwise we would have that $0 < |\gamma| < d$. This contradicts to the fact that

$$\overline{\gamma}^* \cap \langle P_{T,P_0,\xi} \rangle_n \neq \emptyset$$

and $n < N_\xi$. The lemma is proved. □

Now we prove the key lemma of this section.

Lemma 3.17. *Fix a propositional calculus P_0, then the tag system T halts on input ξ if and only if $P_{T,P_0,\xi} \geq P_0$, for all $\xi \in \mathcal{A}^+$.*

Proof. By definition, if the tag system T halts on an input word $\xi \in \mathcal{A}^+$, then $\xi \stackrel{T}{\Longmapsto} \zeta$ for some word $\zeta \in \mathcal{A}^+$ such that $|\zeta| < d$. Since

$$P_T \vdash \overline{\xi} \Rightarrow \overline{\zeta}$$

by Corollary 3.12, the code $\overline{\zeta}$ of ζ is derivable from P_T and $\overline{\xi}$. If we recall that P_{T,P_0} contains the axioms $\overline{\zeta} \to A$ for each $A \in P_0$, we obtain that $P_{T,P_0,\xi} \vdash A$ and so $P_{T,P_0,\xi} \geq P_0$.

Conversely, let $P_{T,P_0,\xi} \geq P_0$. Recall that the formula $x \circ y$ is built up with using a fixed formula $\hat{x}$ as follows:

$$x \circ y = ((\hat{y} \to \hat{y}) \to \hat{y}) \to (\hat{x} \to ((\hat{y} \to \hat{y}) \to \hat{y})).$$

Since $\hat{x}$ is an arbitrary one-variable $\{\to\}$-formula, we may assume that every formula in P_0 is not a substitution instance of $x \circ y$ or $x \circ y \to z$. On the other hand, all formulas having derivations in $P_{T,P_0,\xi}$ of a height less or equal N_ξ is a substitution instances of $x \circ y$ or $x \circ y \to z$ by Lemma 3.16. Hence, if T does not halt on input ξ, then $N_\xi = \infty$ and, therefore, $P_{T,P_0,\xi} \not\geq P_0$. This contradiction completes the proof. □

4 The proof of Theorem 2.10

4.1 Undecidability of recognizing extensions

If **Ext** is decidable for a Σ-calculus P_0, then the following problem is decidable: given a tag system T and a word $\xi \in \mathcal{A}$, determine whether $P_0 \leq P_{T,P_0,\xi}$. By Lemma 3.17, the decidability of the last problem for the calculus P_0 is equivalent to the decidability of the halting problem for the tag system T. Since the halting problem of tag systems is undecidable by Theorem 3.2, this contradiction completes the proof of undecidability of recognizing extensions.

4.2 Undecidability of recognizing completeness

If $\{x \to (y \to x)\} \leq P_0$, then $P_{T,P_0,\xi} \leq P_0$ by Lemmas 3.7 and 3.9. Hence the problem of recognizing completeness of P_0 reduces to the problem of recognizing extensions of P_0, which is undecidable. This completes the proof of the theorem.

5 Conclusion and further research

In this paper, we established the undecidability of the problem of recognizing extensions for all propositional calculus, and the undecidability of the problem of recognizing completeness, as well as axiomatizations, for all propositional calculus whose theorems contain the formula $x \to (y \to x)$. These results were obtained for the signatures containing the symbol of implication $\to$. It is easily shown that the proofs remain valid, with minor changes, if we consider a signature, which does not contain the symbol $\to$, but there is some propositional formula having x, y as sole variables, whose truth-table interpretation is "x implies y".

The other observation is that we can redefine encoding of letters and words by using the formula $x \to (F \to x)$ instead of the formula $x \to (y \to x)$, where F is an arbitrary formula not containing the variable x. If we replace the key formula $x \circ y$ with the following formula

$$((\hat{y} \to \hat{y}) \to \hat{y}) \to (\hat{F}[x] \to ((\hat{y} \to \hat{y}) \to \hat{y})),$$

where $\hat{F}[x]$ is the substitution instance of F by replacing all occurrences of variables with a fixed one-variable formula $\hat{x}$, we obtain the following interesting generalization of Theorem 2.10.

Theorem 5.1. *Fix a propositional formula F not containing the variable x and a propositional calculus $P_0 \geq \{x \to (F \to x)\}$, then **Axm** and **Cmpl** are undecidable for P_0.*

We leave the proof to the reader.

A natural and interesting question arises with respect to this generalization: there is an enumerable set of propositional formulas M for which the condition $[P_0] \cap M \neq \emptyset$ holds if and only if **Axm** and **Cmpl** are undecidable for P_0. Since Gladstone in [7] proved that **Drv** is decidable for every one-variable propositional calculus, it seems to be interesting to consider only formulas containing two or more variables. Theorem 5.1 shows that two-variables formulas are sufficient.

6 Acknowledgement

The author is grateful to Karel Chvalovský for discussion of undecidable problems of propositional calculi investigated by Marcinkowski and useful comments that improved the manuscript.

References

[1] *Bokov G. V.* Completeness problem in the propositional calculus. // Intelligent Systems, vol. 13, no. 1-4, p. 165-182, 2009. (Russian).

[2] *Bokov G. V.* Undecidability of the problem of recognizing axiomatizations for propositional calculi with implication. // Logic Journal of the IGPL, 2015. (Received 24 July 2014).

[3] *Bollman D., Tapia M.* On the recursive unsolvability of the provability of the deduction theorem in partial propositional calculi.. // Notre Dame Journal of Formal Logic, vol. 13, no. 1, p. 124–128, 1972.

[4] *Chagrov A., Zakharyaschev M.* Modal Logic. — Clarendon Press, 1997.

[5] *Davis M.* Computability & unsolvability. — McGraw-Hill, 1958.

[6] *Gladstone M. D.* Some Ways of Constructing a Propositional Calculus of Any Required Degree of Unsolvability. // Transactions of the American Mathematical Society, vol. 118, p. 192-210, 1965.

[7] *Gladstone M. D.* The decidability of one-variable propositional calculi. // Notre Dame Journal of Formal Logic, vol. 20, no. 2, p. 438–450, 1979.

[8] *Harrop R.* On the existence of finite models and decision procedures for propositional calculi. // Mathematical Proceedings of the Cambridge Philosophical Society, vol. 54, no. 1, p. 1–13, 1958.

[9] *Harrop R.* A Relativization Procedure for Propositional Calculi, with an Application to a Generalized Form of Post's Theorem. // Proceedings of the London Mathematical Society, vol. s3-14, no. 4, p. 595-617, 1964.

[10] *Hilbert D., Bernays P.* Grundlagen der Mathematik. — Edward Brothers, 1968.

[11] *Hughes C. E.* Two Variable Implicational Calculi of Prescribed Many-One Degrees of Unsolvability. // Journal of Symbolic Logic, vol. 41, no. 1, p. 39–44, 1976.

[12] *Ihrig A. H.* The Post-Lineal theorems for arbitrary recursively enumerable degrees of unsolvability. // Notre Dame Journal of Formal Logic, vol. 6, no. 1, p. 54–72, 1965.

[13] *Kleene S. C.* Mathematical Logic. — Dover Publications, 2002.

[14] *Kuznetsov A. V.* Undecidability of the general problems of completeness, decidability and equivalence for propositional calculi. // Algebra and Logic, vol. 2, no. 4, p. 47-66, 1963. (Russian).

[15] *Linial S., Post E. L.* Recursive unsolvability of the deducibility, Tarski's comleteness, and independence of axioms problems of the propositional calculus. // Bulletin of the American Mathematical Society, vol. 55, p. 50, 1949.

[16] *Łukasiewicz J.* The shortest axiom of the implicational calculus of propositions. // Proceedings of the Royal Irish Academy. Section A: Mathematical and Physical Sciences, vol. 52, p. 25-33, 1948.

[17] *Marcinkowski J.* A Horn clause that implies an undecidable set of Horn clauses. // Selected papers of the 7th Workshop on Computer Science Logic (CSL '93), vol. 832, p. 223-237, 1994.

[18] *Meredith C.* A single axiom of positive logic. // Journal of Computing Systems, vol. 1, p. 169-170, 1953.

[19] *Minsky M. L.* Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. // Annals of Mathematics, vol. 74, p. 437-455, 1961.

[20] *Minsky M. L.* Computation: Finite and Infinite Machines. — Upper Saddle River, NJ, USA, Prentice-Hall, Inc., 1967.

[21] *Post E. L.* Formal reduction of the general combinatorial decision problem. // American Journal of Mathematics, vol. 65, p. 197-215, 1943.

[22] *Post E. L.* Recursively enumerable sets of positive integers and their decision problems. // Bulletin of the American Mathematical Society, vol. 50, p. 284–316, 1944.

[23] *Ratsa M. F.* Undecidability of the expressibility problem in modal logics. // Mathematical Problems of Cybernetics, vol. 2, p. 71–99, 1989. (Russian).

[24] *Sinaceur H.* Address at the Princeton University bicentennial conference on problems of mathematics (December 17–19, 1946), by Alfred Tarski.. // Bulletin of Symbolic Logic, vol. 6, no. 1, p. 1-44, 2000.

[25] *Singletary W. E.* A complex of problems proposed by Post. // Bulletin of the American Mathematical Society, vol. 70, no. 1, p. 105–109, 1964.

[26] *Singletary W. E.* Results regarding the axiomatization of partial propositional calculi. // Notre Dame Journal of Formal Logic, vol. 9, no. 3, p. 193–211, 1968.

[27] *Tarski A., Corcoran J.* Logic, Semantics, Metamathematics: Papers from 1923 to 1938. — Hackett Publishing Company, Incorporated, 1983.

[28] *Turing A. M.* On computable numbers, with an application to the Entscheidungsproblem. A correction. // Proceedings of the London Mathematical Society. Second Series, vol. 43, p. 544–546, 1937.

[29] *Wang H.* Tag systems and lag systems. // Mathematische Annalen, vol. 152, no. 1, p. 65-74, 1963.

[30] *Yntema M. K.* A detailed argument for the Post-Linial theorems. // Notre Dame Journal of Formal Logic, vol. 5, no. 1, p. 37–50, 1964.

[31] *Zolin E.* Undecidability of the Problem of Recognizing Axiomatizations of Superintuitionistic Propositional Calculi. // Studia Logica, vol. 102, p. 1021–1039, 2014.